Aalto University
School of Science

Benchmarking, Monitoring, Validation and Experimenting for Big Data and Machine Learning Systems

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, *https://rdsea.github.io*

Issues raised from the last discussion

- **Elasticity and concrete tools we can use?**
- **Robustness, Reliability and Resilience are highly connected**
  - how can we model/capture their relations in a system?
- **Runtime attributes are crucial**
  - how can we capture? are existing monitoring tools enough?
- **Workflow and orchestration (another lecture)**
- **R3E as metrics:**
  - abilities/qualities, must be considered throug the DevOps cycle

Learning objectives

- **Able to analyze the role of measurement, monitoring and observability in real-world cases for R3E**

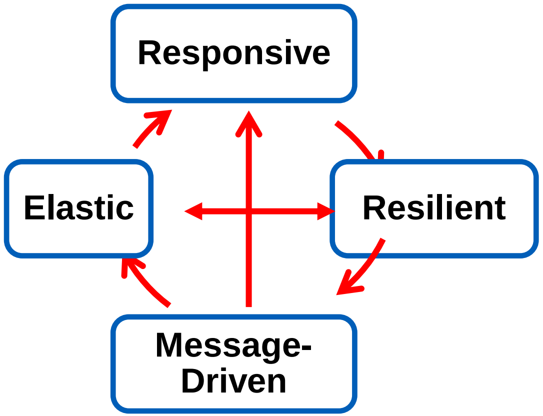
- **Understand and develop methods with key steps and important tools for benchmarking, monitoring, validation and experimenting**

- **Able to apply these methods for big data/ML systems**

The role of measurement, monitoring and observability in real-world cases

Reactive systems – an architectural style for R3E?

Reactive systems



Source: https://www.reactivemanifesto.org/

For R3E abilities, big data/ML systems can be designed with ”reactive systems” principles:

- **Responsive:**
  - capture and respond to quality indicators, QoA
- **Resilient:**
  - deal within failures
- **Elastic:**
  - deal with different workload and quality of analytics
- **Message-driven:**
  - allow loosely coupling, isolation, asynchronous

Development vs Runtime activities

Design, test and benchmark R3E

- R3E for individual components
- model/capture complex dependencies
- design logs, metrics and traces for capturing states and complex dependencies

Monitoring/Observability and Runtime adaptation

- runtime monitoring and observability
- states, performance and failure analytics
- runtime controls (constraints, rules, actions)

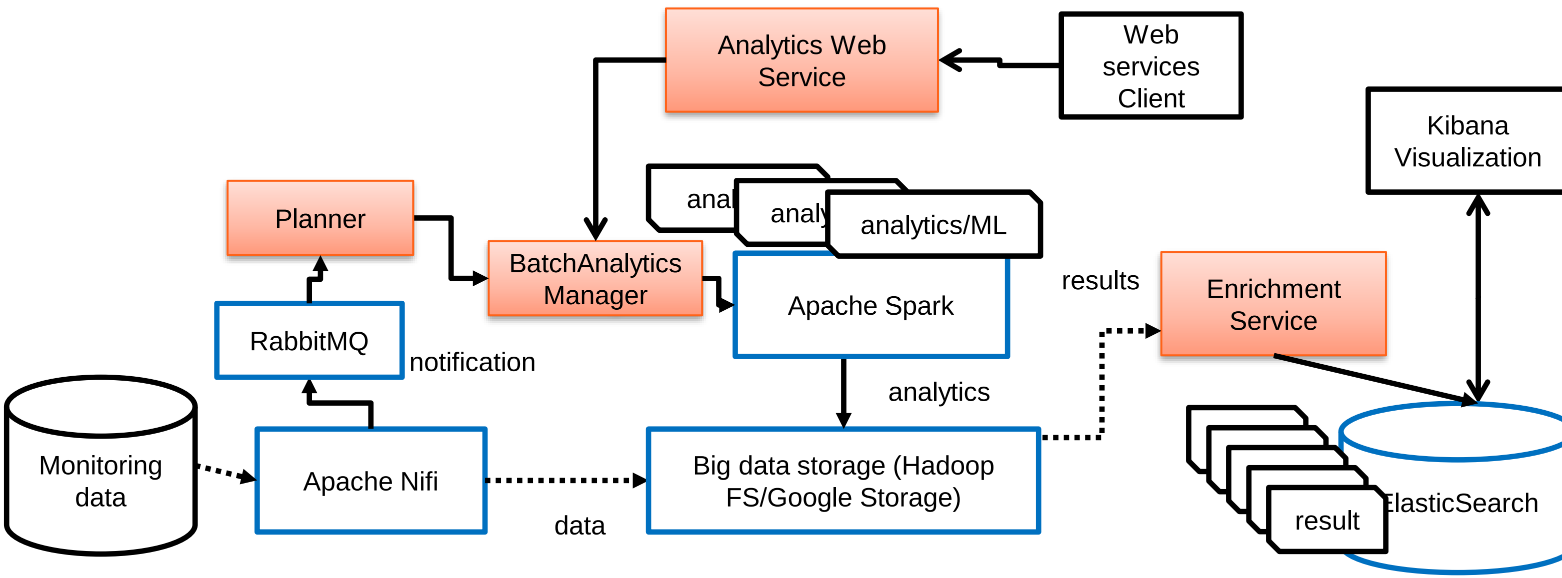
Measurement, Monitoring and Observability for R3E

- **Instrumentation and sampling**
  - instrumentation: insert probes into systems so that you can measure system behaviors directly or produce logs
  - sampling: use components to sample system behaviors
- **Monitoring**
  - perform sampling or measurements; store and share measurements, metrics, and logs; show what happening
- **Observability**
  - evaluate and interpret measurements for specific contexts
  - understand and explain the systems states, dependencies, etc.

Aalto University
School of Science

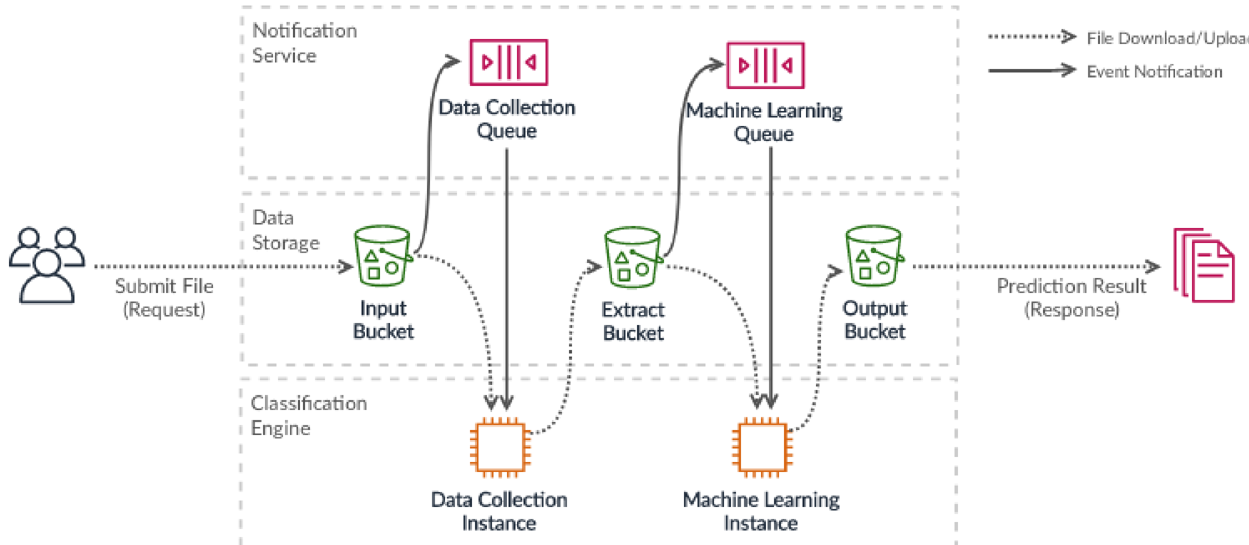
Why is it challenging to do monitoring/observability for today‘s big data/ML systems?

Discussion: Monitoring/Observability and R3E



Source: Linh Truong, „*I & A Big Data Platform*“, Industrial Work, Not published, 2018

Discussion: Monitoring/Observability and R3E

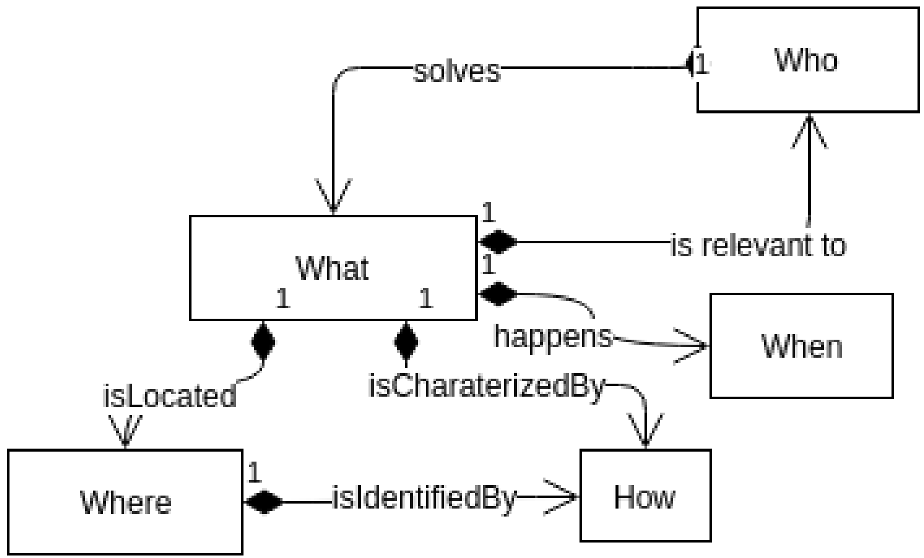


Source: Minjung Ryu, „*Machine Learning-based Classification System for Building Information Models* ", Aalto CS Master thesis, 2020

Methods

What/Which, Where, When, Who and How

Understand W4H aspects for analytics of big data/ML systems



Key steps – What/Which

- **Understand and identify indicators/metrics characterizing big data/ML systems**
- **Common metrics but you might have some specific ones or have different relevance for your metrics**
- **Most critical problems are due to complex dependencies that are not common**
- **For which purposes?**
  - SRE, benchmarking, Test-Driven Development (TDD)

Key steps – Where and When

- **Where: as a “ space” dimension**
  - Tightly coupled or isolated/loosely coupled
  - Identify where
    - software/system layers, components and systems boundaries
    - dependencies among components
- **When: as a „time“ dimension**
  - Design, Test/Training, Runtime (DevOps)

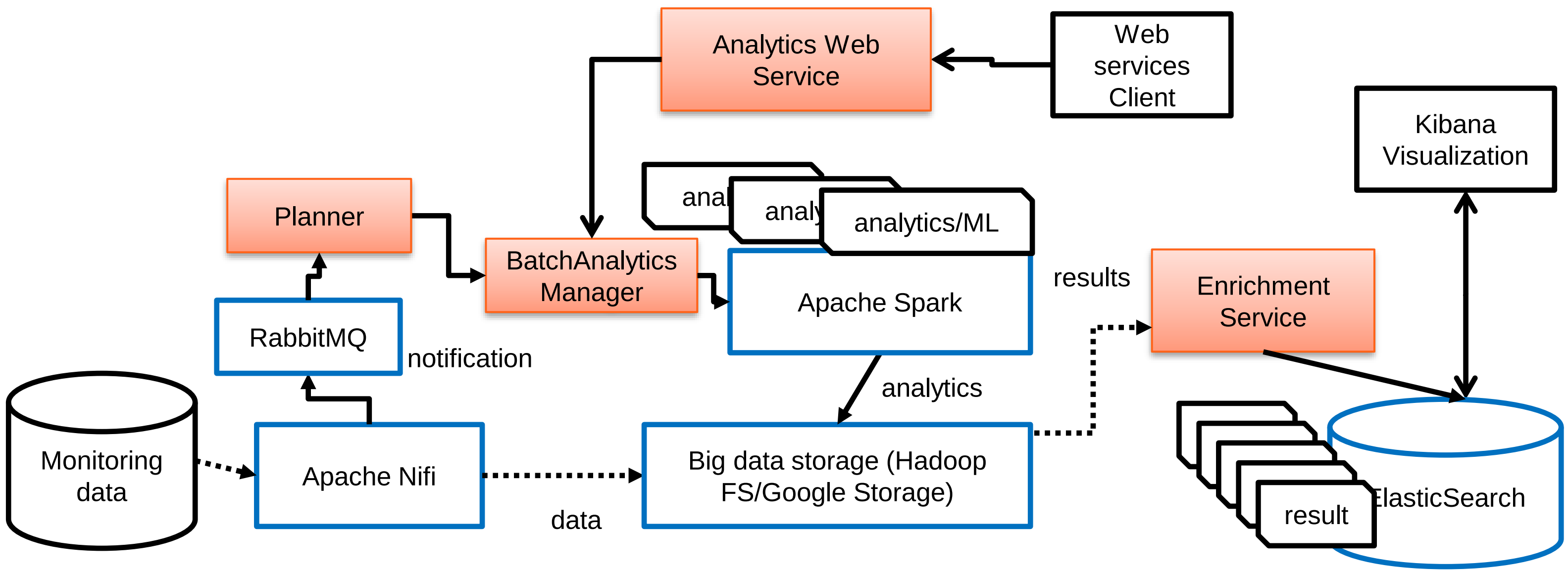
Key steps - How

- **Characterize dependencies among components**
- **Select tools for capturing metrics**
- **Understand what kind of changes/designs we must do**
- **Do monitor and analysis**
- **Integrate many types of data for analytics**

Apply W4H for dealing with benchmarking, monitoring, validation and experimenting

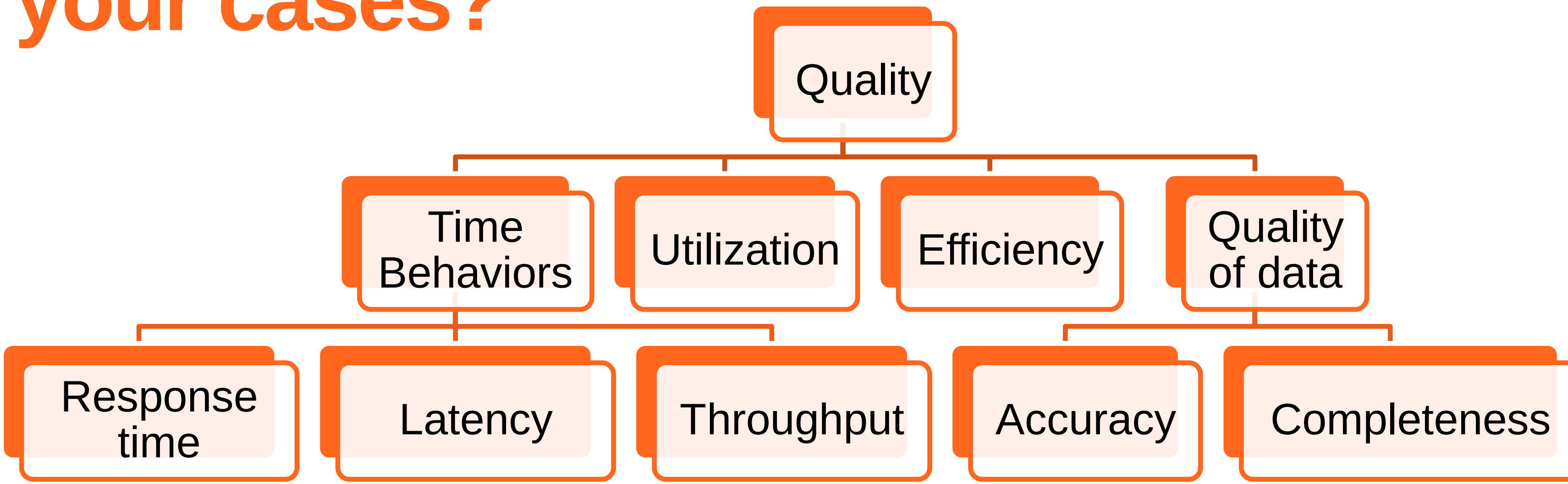
- **Determines clearly system boundaries**
  - the system under study, the system used to judge, and the environment
- **Understands dependencies**
  - among components in distributed big data/ML systems in distributed computing platforms
  - single layer as well as cross-layered dependencies
- **Determines types of metrics and failures and break down problems along the dependency path (how)**

Boundaries and dependencies?



Source: Linh Truong, „*I & A Big Data Platform*“, Industrial Work, Not published, 2018

What are the most critical metrics for your cases?



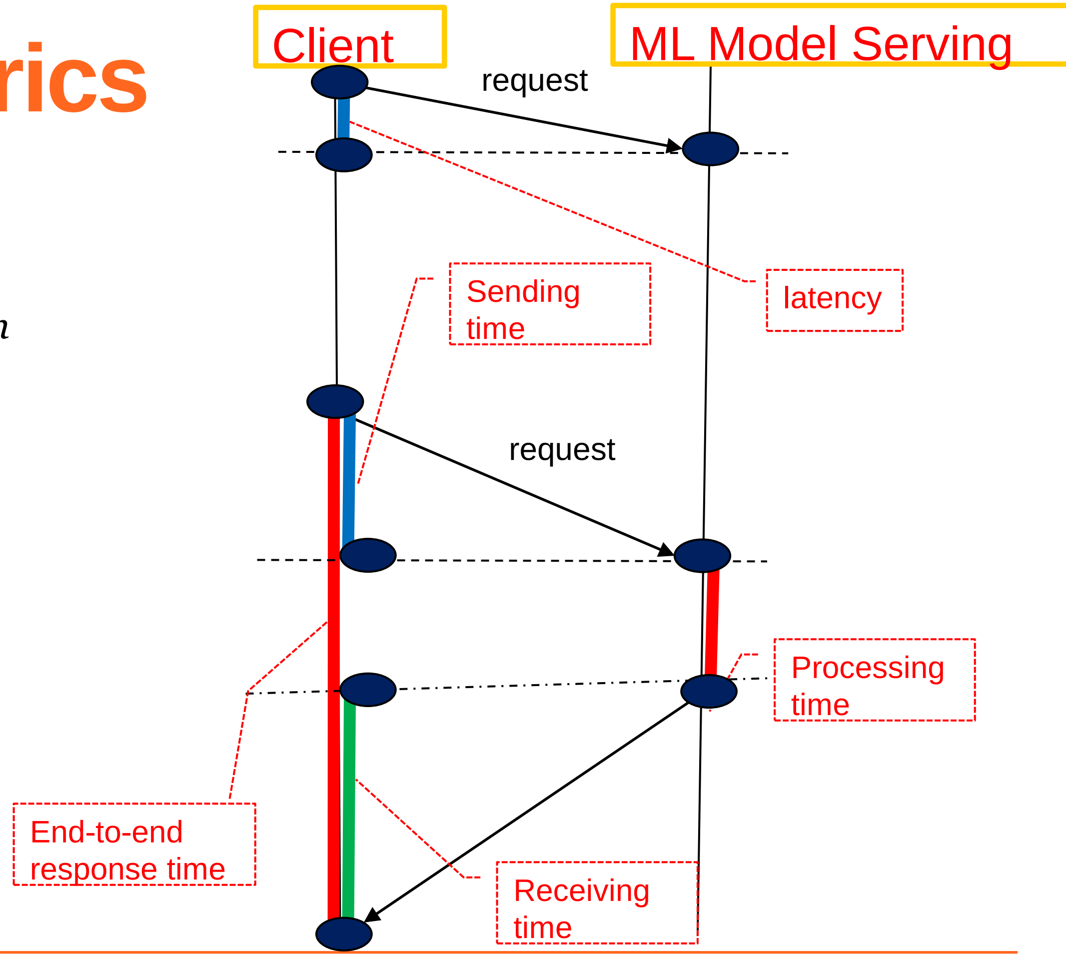
Industry view: https://guidingmetrics.com/content/cloud-services-industrys-10-most-critical-metrics/
NIST: https://www.nist.gov/sites/default/files/documents/itl/cloud/RATAX-CloudServiceMetricsDescription-DRAFT-20141111.pdf

Contradiction/Tradeoffs between Efficiency versus Resiliency

Common performance metrics

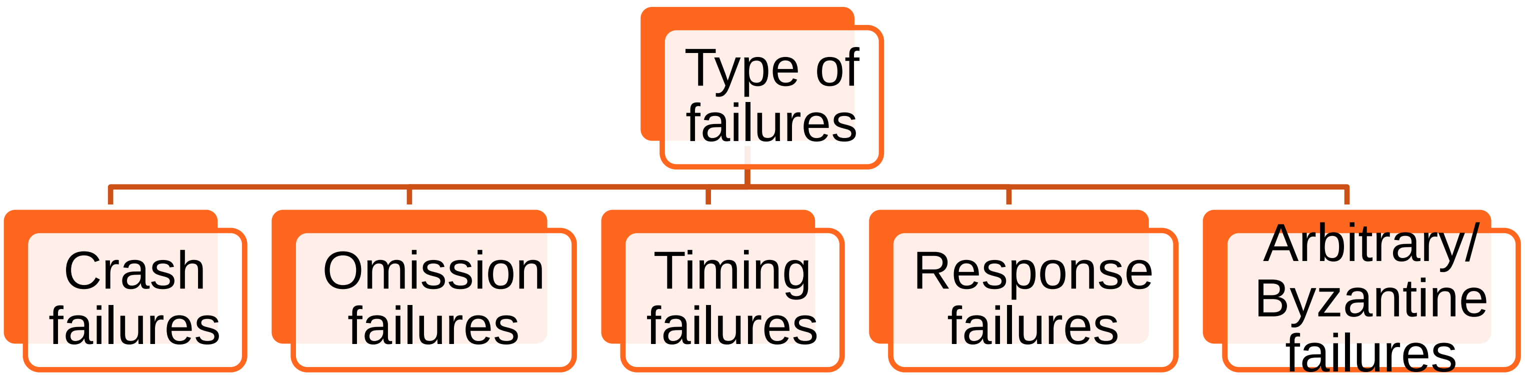
- **Timing behaviors**
  - Communication
    - *Latency/Transfer time*
    - *Data transfer rate, bandwidth*
  - Processing
    - *Response time*
    - *Throughput*
- **Utilization**
  - Network utilization
  - CPU utilization
  - Service utilization
- **Efficiency/Scalability**
  - Concurrent Executions

Examples



BUT ARE THEY ENOUGH?

Types of Failure



But unforeseen failures cannot be determined in advance → design for handling failure

Data Quality

- Completeness
- Timeliness
- Currency
- Validity
- Format
- Accuracy
- Data Drift

Metrics for ML models

- **Concept drift**
  - (https://en.wikipedia.org/wiki/Concept_drift)
- **Confusion matrix**
- **Accuracy**
- **Loss**
- **True positive rate**
- **False positive rate**
- **F1 Score/F-measure**
- **Etc.**

(see https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234)

Aalto University
School of Science

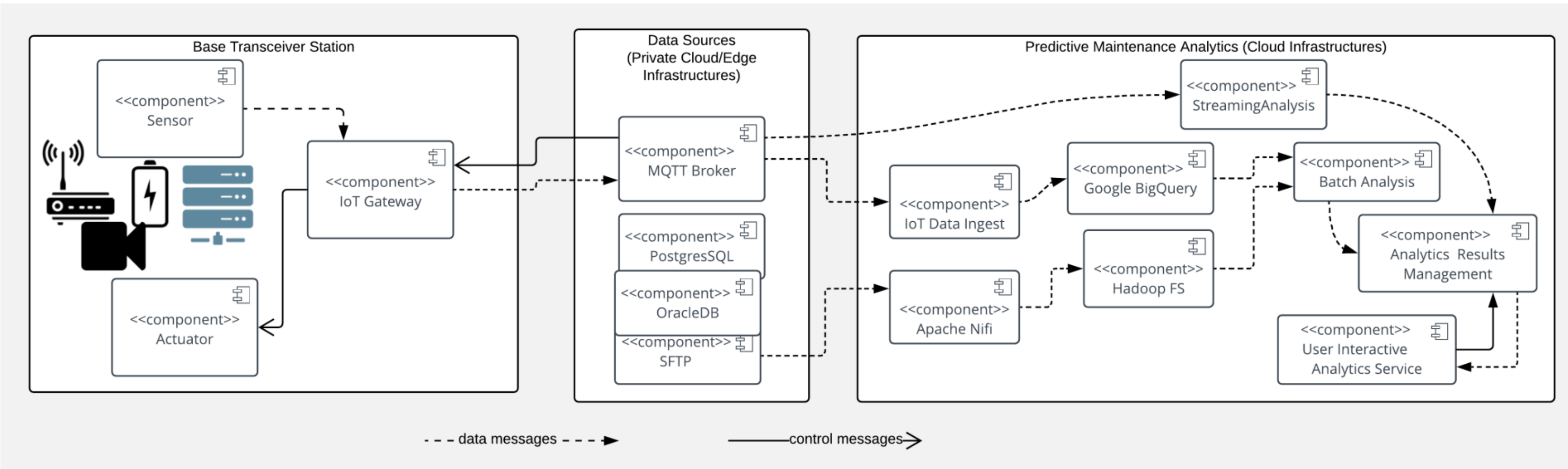
Who can (and how to) establish relations among them?

Benchmarking

- **Benchmark: for comparing big data/ML systems w.r.t. selected (standard/common) workloads**
- **Where to be benchmarked**
  - benchmark individual subsystems: message brokers and data ingestion, databases and ingestion/query, data processing, serving platform
- **What to be benchmarked**
  - data ingestion throughput, processing throughput and time, component CPU and memory
  - training and inferencing time and accuracy

Benchmarking

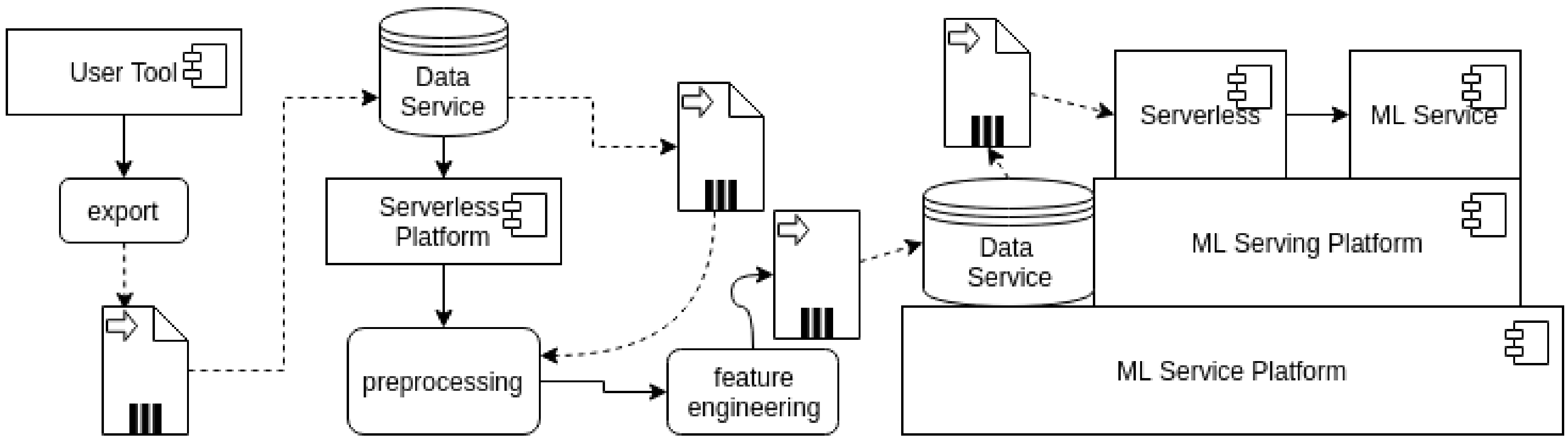
What we should do for a big data system?



Check: https://www.sciencedirect.com/science/article/pii/S0140366419312344
https://www.benchcouncil.org/BigDataBench/

Benchmarking

If you have an end-to-end ML system, does it make sense to benchmark the whole system? What should we do?



Benchmarking - ML

Examples:

Benchmark	Dataset	Quality Target	Reference Implementation Model
Image classification	ImageNet (224x224)	75.9% Top-1 Accuracy	Resnet-50 v1.5
Object detection (light weight)	COCO 2017	23% mAP	SSD-ResNet34
Object detection (heavy weight)	COCO 2017	0.377 Box min AP, 0.339 Mask min AP	Mask R-CNN
Translation (recurrent)	WMT English-German	24.0 BLEU	GMNT
Translation (non-recurrent)	WMT English-German	25.0 BLEU	Transformer
Recommendation	Undergoing modification		
Reinforcement learning	N/A	Pre-trained checkpoint	Mini Go

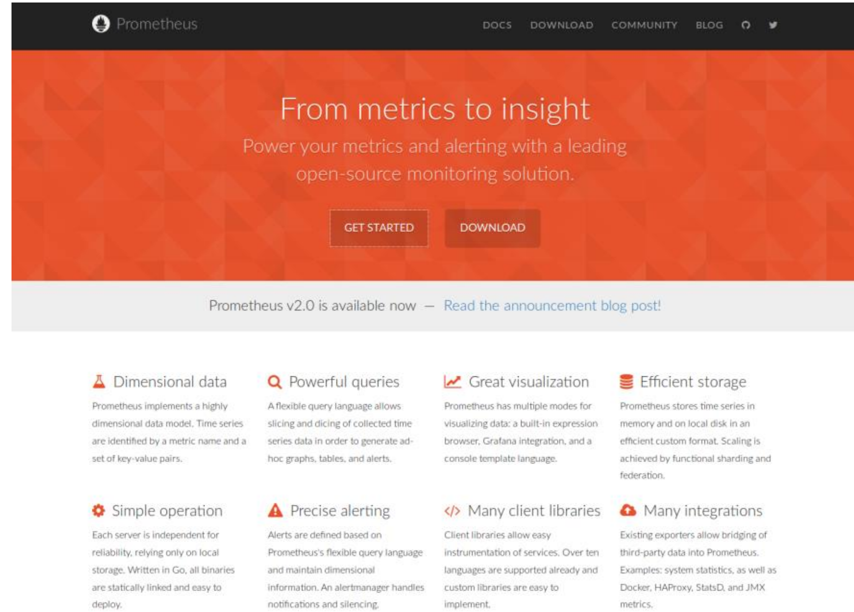
Source: https://mlperf.org/training-overview

Also check: https://www.benchcouncil.org/AIBench/index.html

Service/Infrastructure Monitoring Tools

There are many powerful tools!

But only low-level, well-identified monitoring information (infrastructures): pre-defined metrics exposed through interfaces with push/pull mechanism



From: https://prometheus.io/

Instrumentation for Observability

Code instrumentation and logs: for many metrics that cannot be monitored from the outside of the component

→

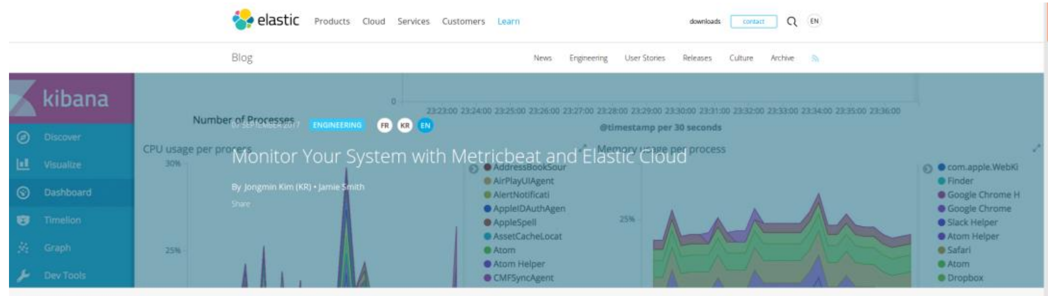
the developer can instrument the code to capture metrics

From: https://www.fluentd.org/

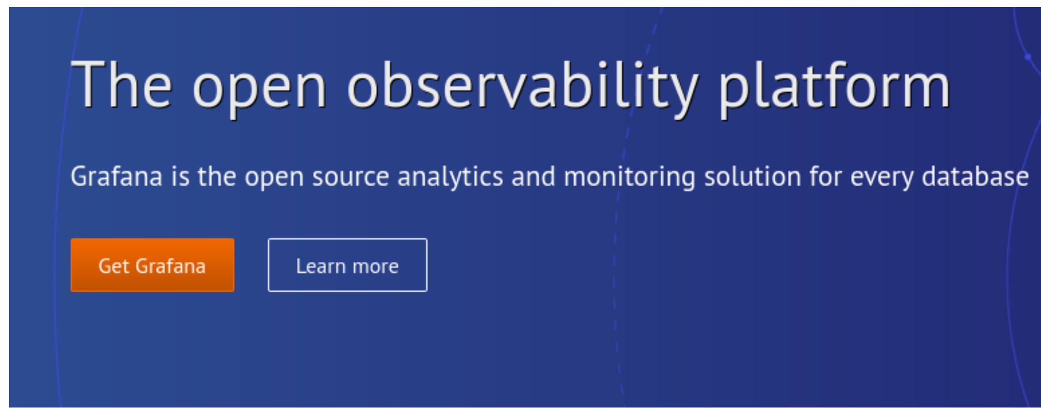
Visualization

Metrics and Visualization

- **Easy to visualize many types of metrics**
- **But only you can specify, define and map to your applications**



https://www.elastic.co/products/kibana



https://grafana.com/

Aalto University
School of Science

What is your approach to capture “application/system complex dependencies and states”

Data & Model Validation/Analysis

- **Not just performance but also „inclusion and fairness“**
- **By humans or by software?**
  - Which one can be done by humans and by software?
- **Data validation tools are very diverse, depending on the frameworks and data**
  - E.g., Tensors Flows: https://www.tensorflow.org/tfx/guide/tfdv

Data & Model Validation/Analysis

- **Model Analysis:**
  - E.g., https://www.tensorflow.org/tfx/model_analysis/

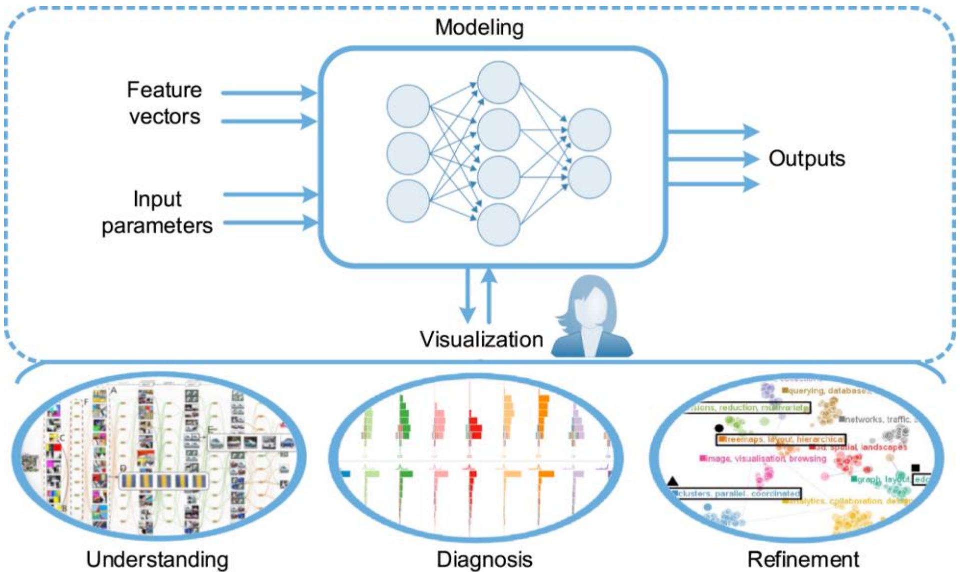
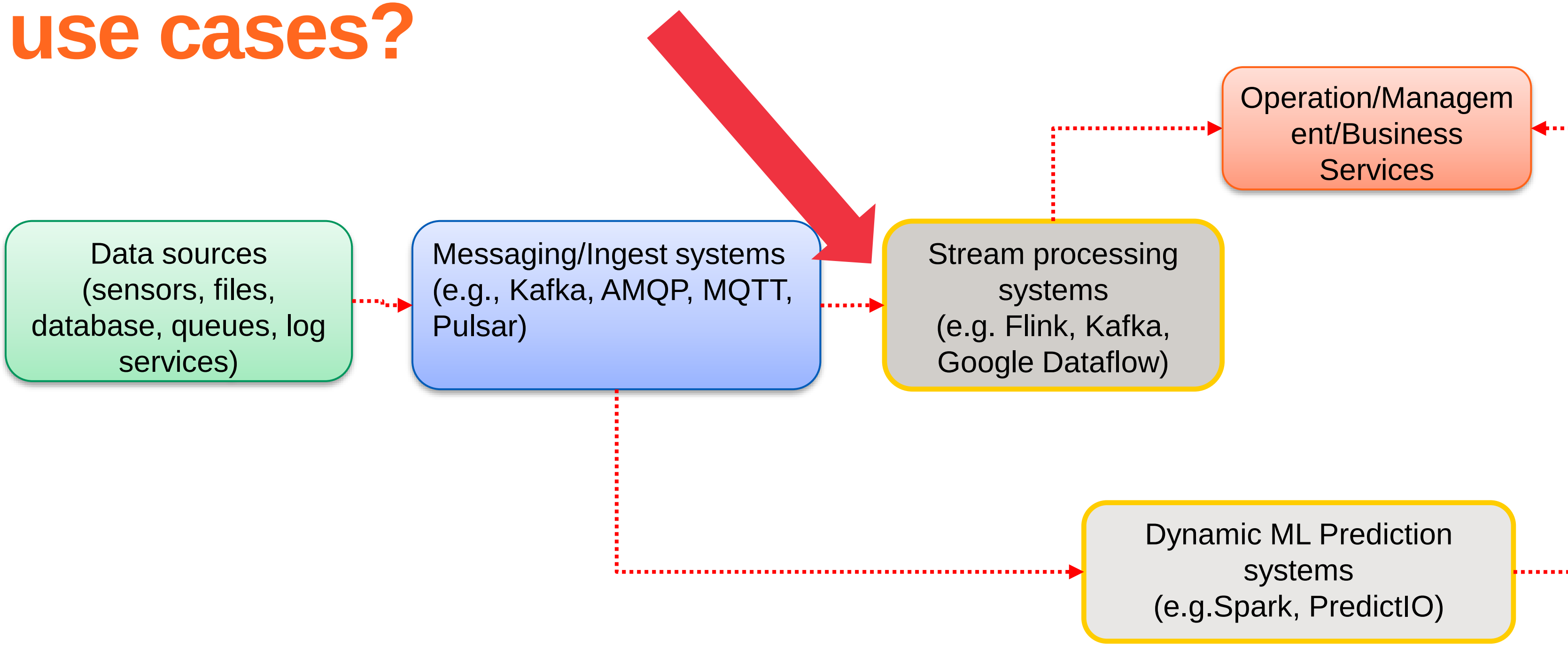


Figure source: Shixia Liu, Xiting Wang, Mengchen Liu, Jun Zhu,
Towards better analysis of machine learning models: A visual analytics perspective,
Visual Informatics, Volume 1, Issue 1, 2017,
https://doi.org/10.1016/j.visinf.2017.01.006.

Can we validate data on-the-fly? For which use cases?



Currently, HILSA is under the development in our team for this purpose

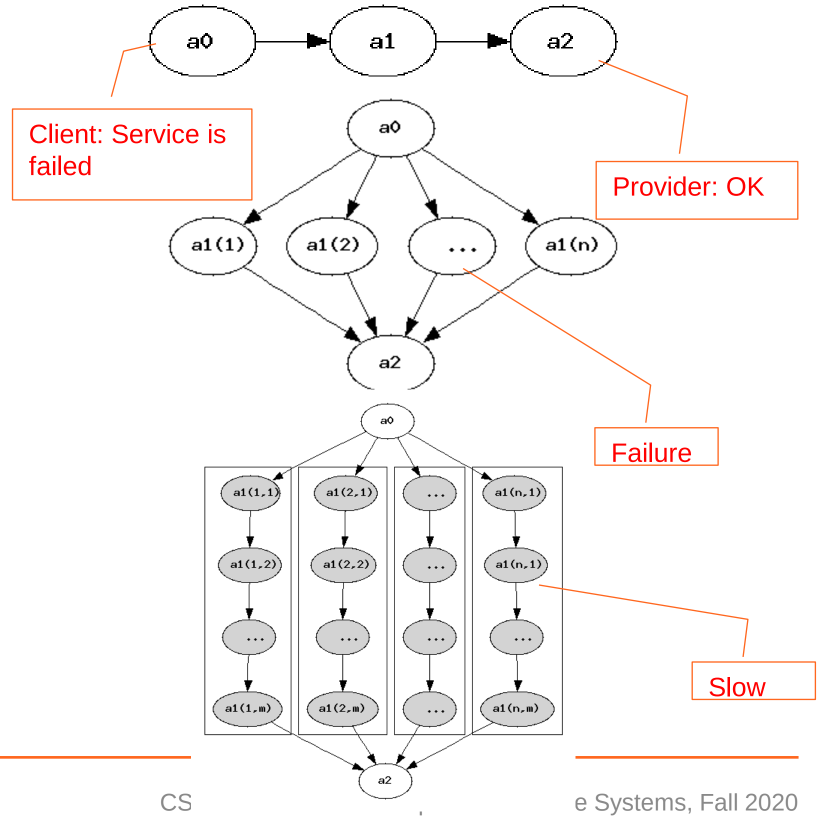
Observability

- **To monitor and understand the system as whole, end-to-end**
  - Every component must be monitored
  - Dependencies/interactions must be captured
  - Metrics, logs, tracing, etc are needed to be integrated
- **Understand the states and behaviors of the whole systems**
- **Complex problems in big data/ML systems as these systems**
  - large-scale number of microservices in large-scale virtualized infrastructures
  - multi-dimensional states (code, models and data)

Do we understand the structure of big data/ML application

- **Composable method**
  - divide a complex structure into basic common structures
  - each basic structure has different ways to analyze specific failures/metrics
- **Interpretation based on context/view**
  - client view or service provider view?
  - conformity versus specific requirement assessment

Dependency Structure



Support an end-to-end view or not

- **End-to-end reflects the entire system**
  - e.g., data reliability: from sensors to the final analytics/inference results
  - what if the developer/provider cannot support end-to-end?
- **The user expects end-to-end R3E**
  - e.g., specified in the expected accuracy
- **Providers/operators want to guarantee end-to-end quality**
  - need to monitor different parts, each has subsystems/components
  - coordination-aware assurance, e.g., using elasticity

Techniques for addressing problems in different system/software layers

- **Immutable infrastructures: containers and orchestration**
  - shared nothing for isolation, redundancy elasticity, auto-recovery
- **Services:**
  - redundancy, data/function sharding, microservices for isolation, elasticity/autoscaling-based, stateless
- **Tasks:**
  - fault-tolerance, retries, delegation
- **Interactions/Requests**
  - service-based, well-defined protocols for isolation, asynchrononous modes for isolation, elasticity, handling cascading failures

Aalto University
School of Science

Example:
The goal is to avoid (cascading) failures in serving requests which is a common problem

Resilience techniques have to be applied in many places (due to many types of request)

Example: resilience implementation strategies for request handling

- **Component/service replication**
  - multiple instances, both data and function sharding
- **Component/service Isolation**
  - asynchronous communications among services, microservices (virtualization/containers), share nothing infrastructural design, failure isolation, well-defined protocols
- **Component/service function delegation**
  - hand over the tasks to other components through task distribution/orchestration via workflows, queues and serverless

Example: resilience implementation strategies for request handling

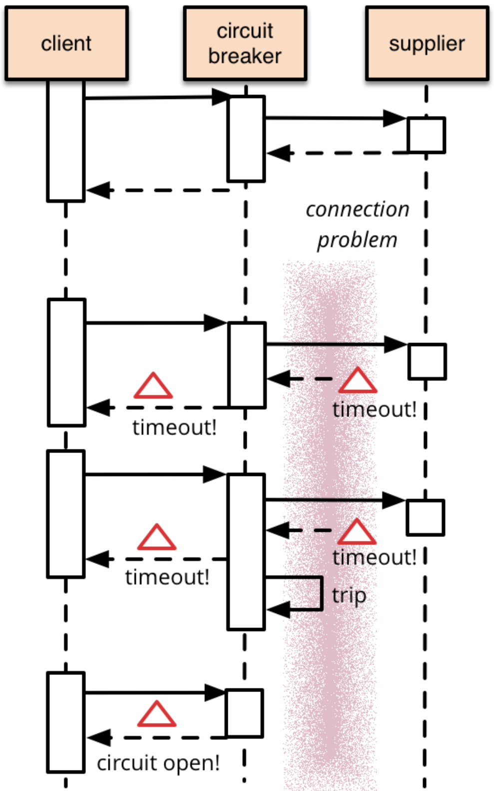
- **Throttling Pattern**
- **Circuit breaker pattern**
- **Queue-based Load Levelling Pattern**
  - https://docs.microsoft.com/en-us/azure/architecture/patterns/queue-based-load-leveling
- **Retry Pattern: exponential backoff**
  - https://cloud.google.com/iot/docs/how-tos/exponential-backoff
- **Many implementation guides and tools, e.g.**
  - https://github.com/resilience4j/resilience4j

Circuit breaker pattern

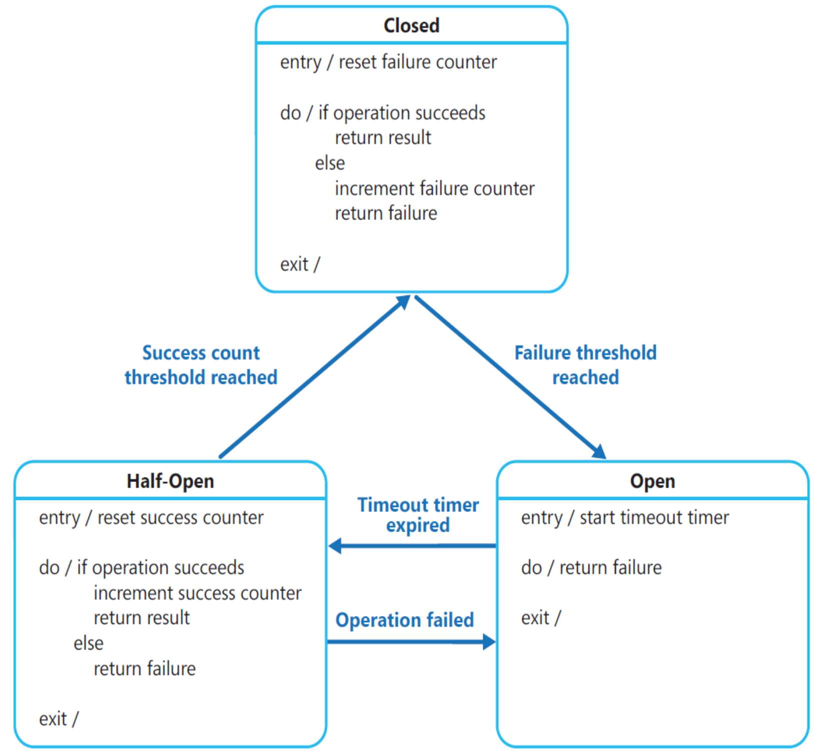
- *What if service operations fail due to unexpected problems or cascade failures (e.g. busy → timeout)*
  - Let the client retry and serve their requests may not be good

→ Circuit breaker pattern prevents clients to retry an operation that would likely fail anyway and to detect when the operation failure is resolved.

Circuit breaker patterm



Source: http://martinfowler.com/bliki/CircuitBreaker.html



Source: https://msdn.microsoft.com/en-us/library/dn589784.aspx

Experiment management: how do we manage important information for ML model?

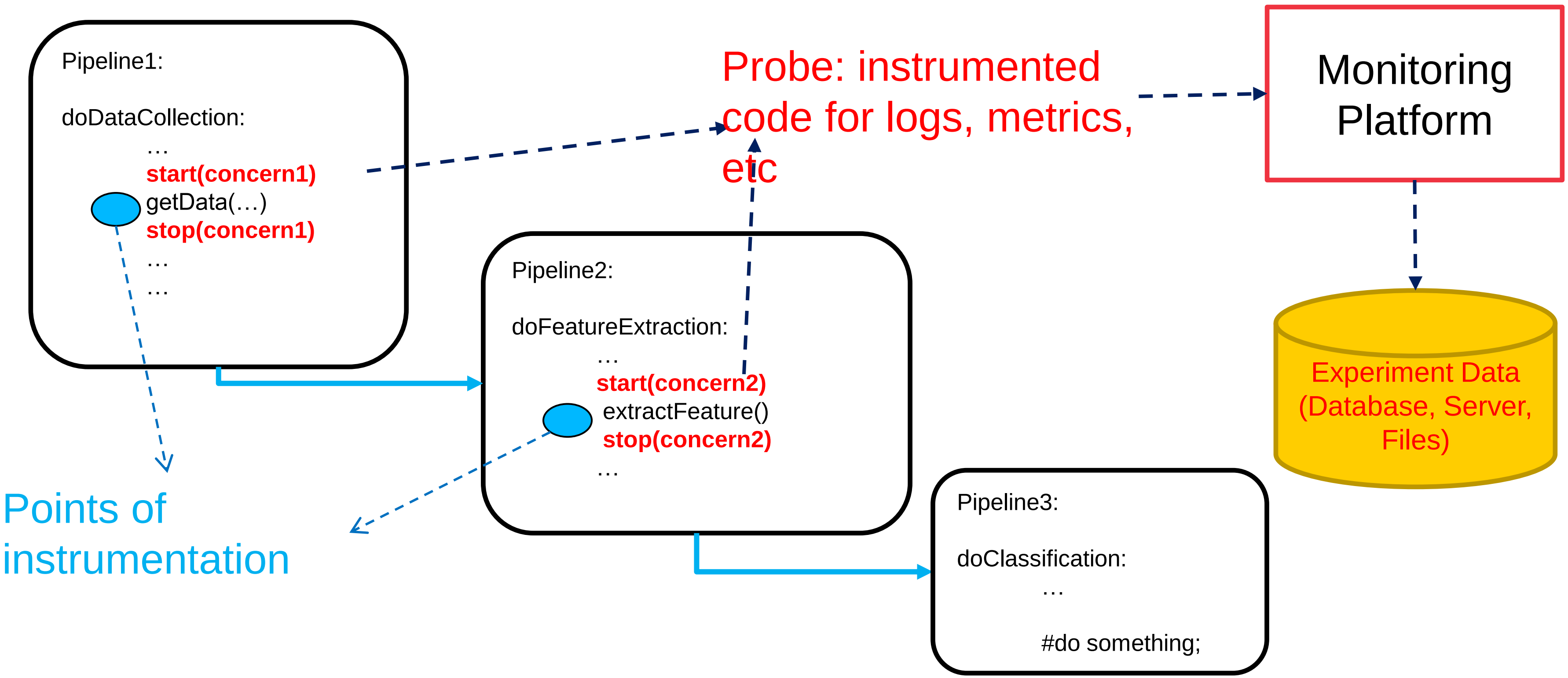
Problems

- **We need to run many experiments**
  - testability/observability purposes: figure out suitable configurations
  - how does this help to understand and support R3E?
- **Experiment management**
  - known domain and well-known books (e.g., “Design and Analysis of Experiments” by Douglas C. Montgomery)
  - principles: capturing various configurations
  - how does it work in big data and ML?
- **What do we need?**
  - tools/frameworks for tracking experiments

Notions

- **A single run/trial**
  - inputs, results, required software artefacts
  - computing resources, logs/metrics
- **Experiment**
  - a collection of runs/trials/executions gathered in a specific context
- **Steps**
  - parameterization: generate different parameters
  - deployment: prepare suitable environments
  - execution: run and collect metrics
  - analysis and sharing: analyze experiment data

Experiment tracking



But remember it is very large system! Which tools can we use?

Examples

- **Experiment in Azure ML SDK**
  - https://docs.microsoft.com/en-us/python/api/overview/azure/ml/?view=azure-ml-py#experiment
- **MLFlows**
  - https://mlflow.org/
- **Kubeflows**
  - https://www.kubeflow.org/docs/pipelines/overview/concepts/
- **DVC**
  - https://dvc.org/

Examples: MLFlow APIs

- **Experiment**

  `mflow.start_run()/end_run()`
- **Logs/metrics collection**

  `mflow.set_tag()`

  `mflow.log_*()`
- **Tracking data management**
  - Local files, Databases, HTTP server, Databrick logs

(follow our hands-on tutorial)

Applying W4H methods for identifying incidents in big data systems: an example

Incidents in cloud-based big data system

If you monitor alarms in a station and see this

What could be happened?

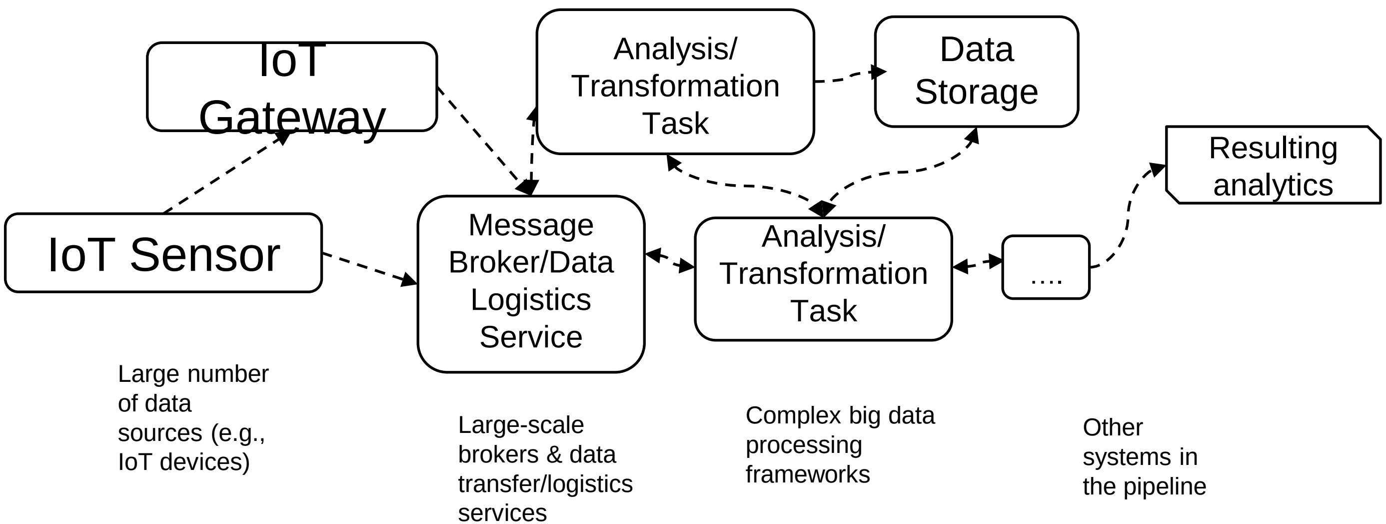
How to deal wit 200K stations?

Steps: Incident monitoring and analytics

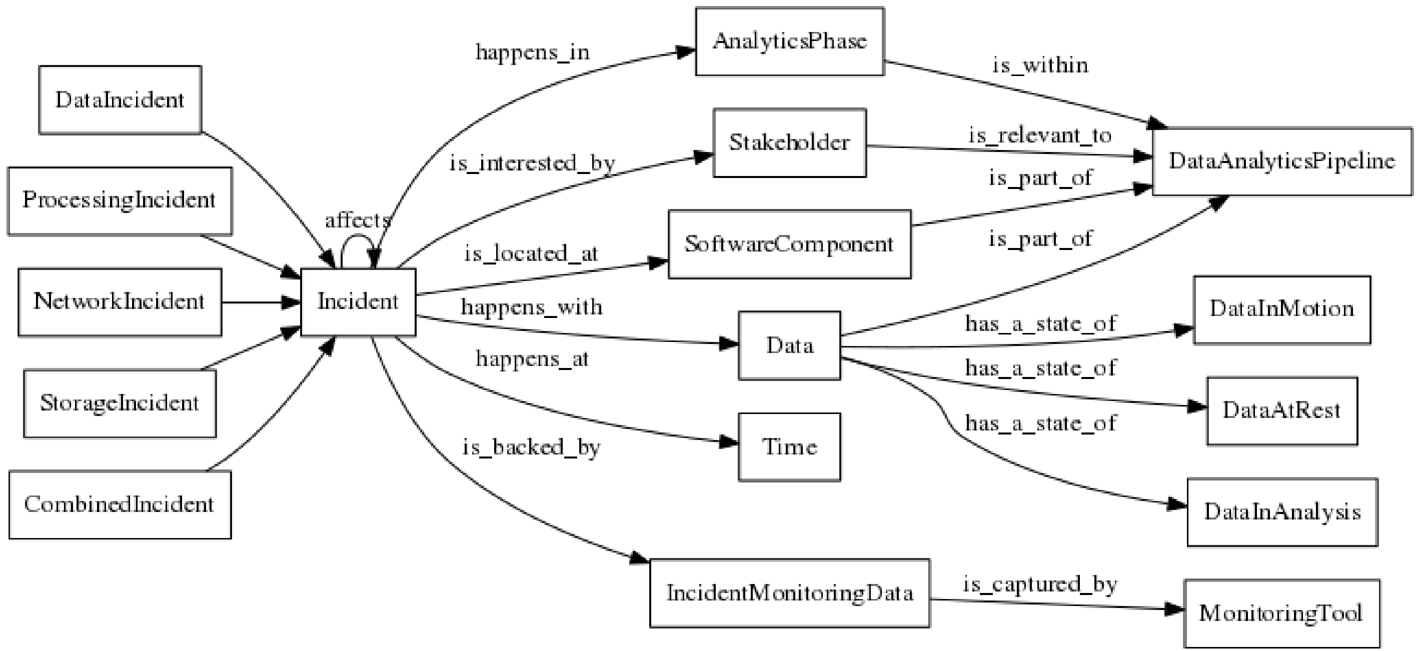
- **Classification of incidents:**
  - to quantify incidents and identify possible data sources, monitoring techniques and analytics.
- **Measurement/Instrumentation:**
  - to provide mechanisms for measurement and data collection for incidents.
- **Incident analytics/observability:**
  - to find out the root cause and dependencies of incidents.

What, when, where and how for incidents

Too complex with many types of software. Can we have a simplified taxonomy for mapping incidents?



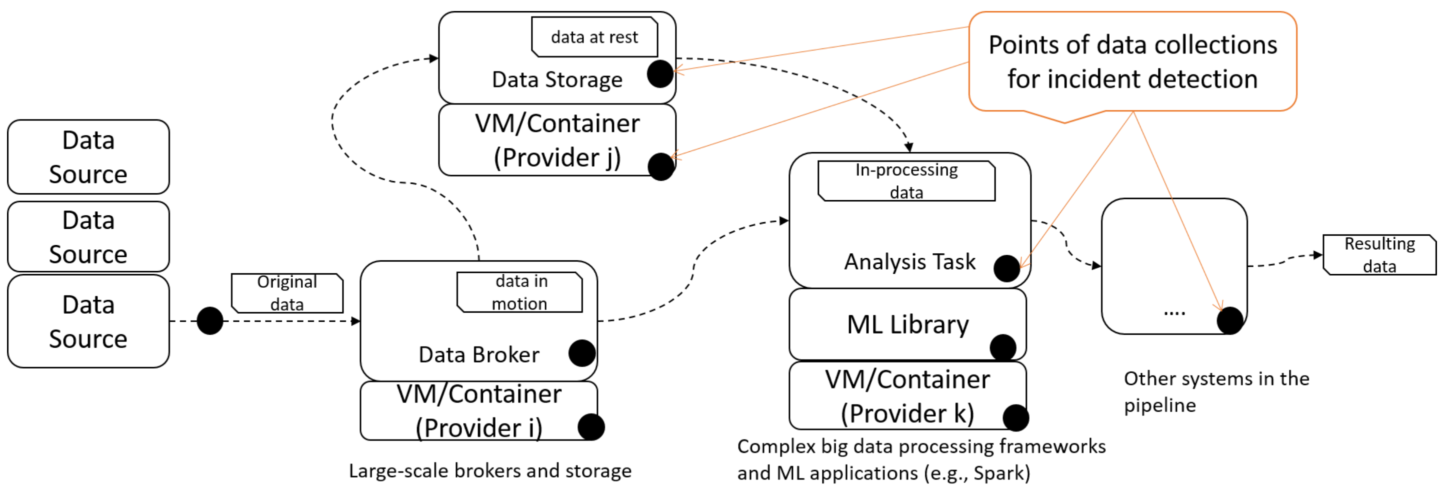
Examples: classification of incidents



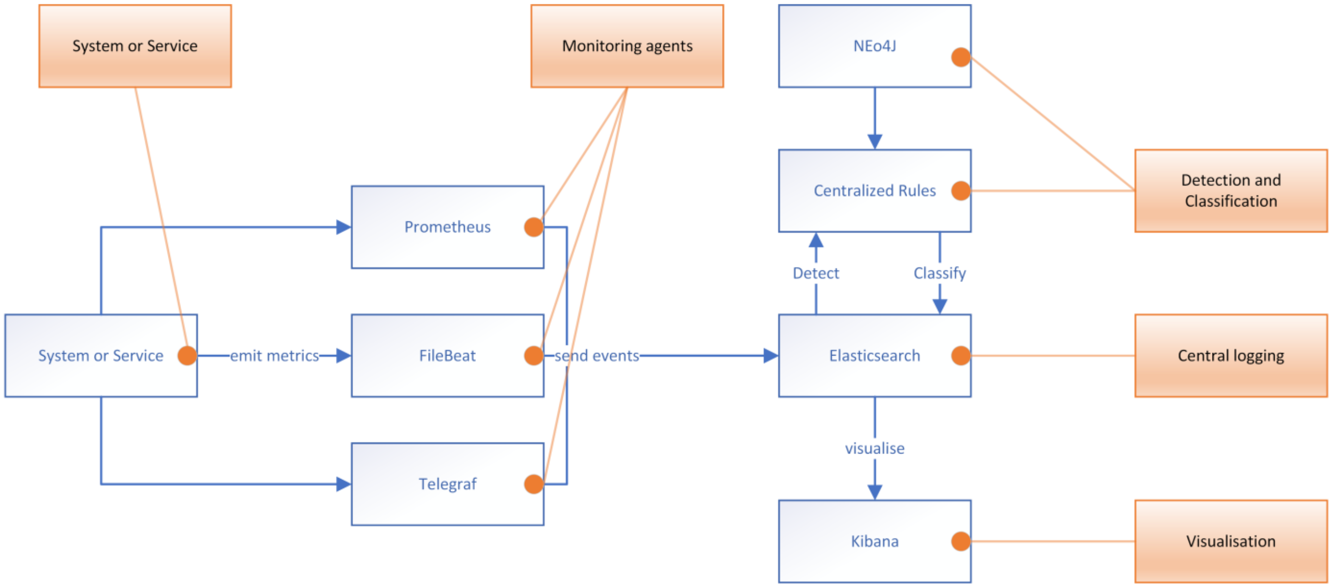
Hong-Linh Truong, Manfred Halper, **Characterizing Incidents in Cloud-based IoT Data Analytics**,, The 42nd IEEE International Conference on Computers, Software & Applications Tokyo, Japan, July 23-27, 2018.

Points of instrumentation for gathering data for incident analytics

Capture monitoring data to analyze and solve incidents, especially incidents related to data quality, across subsystems in ensembles to achieve quality of results



Integration monitoring and instrumentation for observability



First outcome: https://github.com/rdsea/bigdataincidentanalytics

What should we do in the next step: reasoning of incidents?

Aalto University
School of Science

What about applying this approach for an end-to-end ML system?

Study log 2

Describe one big data/ML pipeline that you are familiar with and explain your thoughts on how would you support "benchmarking", "monitoring", "observability", "validation", "experimenting" or "design pattern" for testing/implementing R3E aspects

- Is enough to focus on 1 pipeline and 1 aspect
- Be concrete, e.g., with metrics and possible tools
- Analyze if things can be done easily or where are the challenges that might be interesting for further investigation

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io